VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2012 Alexander Dračka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Alexander Dračka**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Stratech Software s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. RNDr. Petr Šaloun, Ph.D.**

Konzultant bakalářské práce: Ing. Petr Havrlant

Datum zadání: 18.11.2011
Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 25.5.2012

.....

Podpis

Ďakujem firme Stratech Software s.r.o. za príležitosť byť súčasťou softvérového tímu a za možnosť vykonávať odbornú prax. Ďakujem Ing. Petrovi Havrlantovi a Petrovi Bystrzyckému za zaškolenie do vyvíjaného produktu a neochvejnú trpezlivosť pri pomoci v pridelených úlohách. Ďakujem doc. RNDr. Petrovi Šalounovi, Ph.D za odborný dohľad pri písaní bakalárskej práce.

Abstrakt

Z obsahu bakalárskej práce sa dozviete o mojej individuálnej odbornej praxi v softvérovej firme Stratech Software s.r.o. V úvodnej časti práce vám predstavím profil firmy a bližšie informácie o českej pobočke. Dozviete sa podrobné informácie o danom produkte a platforme, na ktorej je vyvíjaný. Podrobne popíšem technológie, ktoré som používal pri práci na projekte. Ďalej rozoberiem vybrané úlohy, na ktorých som pracoval počas praxe. Na záver zhodnotím teoretické a praktické prínosy a nedostatky odhalené počas odbornej praxe.

Kľúčové slová:

.NET, Silverlight, C#

Abstract

In this bachelor thesis I am writing about my experiences during my individual professional practice at the software company Stratech Software Ltd.. In the introductory part of the thesis I am going to introduce the company profile as well as its czech branch office. I am going to provide a detailed describtion of the product in development, its platform and also the technologies I used while working on the project. Furthermore, I am going to analyze the individual assignments I was working during my practice. Finally, I am goint to evaluate the theoretical and practical benefits of this kind of practice as well as point out the lack of some technical knowledge on my behalf that I discovered during the practice.

Key words:

.NET, Silverlight, C#

Zoznam použitých skratiek a symbolov

.NET	- Software framework (súbor technológií tvoriacich celú platformu)
Framework	- Software/Hardware framework (vývojová platforma)
C#	- Programming language (programovací jazyk)
GUI	- Graphical User Interface (grafické používateľské rozhranie)
UI	- User Interface (používateľské rozhranie)
MVVM	- Model-View-ViewModel design pattern(návrhový vzor Model-View-ViewModel)
MEF	- Managed Extensibility Framework (platforma pre modulárne a rozšíriteľné aplikácie)
WCF	- Windows Communications Foundation (platforma pre distribuované aplikácie)
WPF	- Windows Presentation Foundation (technológia pre tvorbu RIA)
RIA	- Rich Internet Aplication (webové aplikácie s pokročilou funkcionalitou)
Prism	- Composite pattern (súbor návrhových vzorov)
SCRUM	- Agile software development process (aktívna metodika pre vývoj softvéru)
XAML	- Extensible Application Markup Language (doplnkové programovacie rozhranie WPF)
TFS2010	- Microsoft Team Foundation Server 2010 (nástroj pre tímovú spoluprácu)
ScrollBar	- panel na horizontálny alebo vertikálny pohyb v okne
pattern	- vzor (návrhový, architektonický apod.)
user–friendly	- používateľsky prívetivé (napr. prostredie aplikácie apod.)
code-behind	-funkčná vrstva doplňujúca návrh jazyka XAML

Obsah

1. Úvod

V úvode by som vám rád predstavil, čo popisuje táto bakalárska práca. Dočítate sa podrobnosti o vykonávaní odbornej praxe v softvérovej firme z pohľadu študenta informatiky. Najprv vám ponúknem stručný profil firmy, pole pôsobnosti na trhu, profil českej pobočky a moje pracovné zaradenie vo vývojovom procese.

V ďalšej časti rozoberiem a popíšem platformu Prodigy a jej architektúru, pod ktorou sa produkt RD-Process vyvíja. Informujem vás o tomto produkte, aké nadobudne využitie na trhu, jeho možnosti, technickú funkcionalitu a popíšem jednotlivé moduly. Súčasťou odbornej praxe je priame zapojenie študenta do vývoja softvéru. Oboznámim vás s úlohami, ktoré som vykonával, ich časovú náročnosť a rozsah v rámci projektu.

Na záver ponúknem hodnotenie svojich výsledkov na základe nadobudnutých odborných znalostí. Popíšem teoretické a praktické nedostatky, ktoré sa objavili pri práci na riešených úlohách.

2. Profil firmy Stratech Software

Stratech Software s.r.o. je holandská spoločnosť s medzinárodným pôsobením, ktorá bola založená v roku 1989. Dcérske pobočky sa nachádzajú vo Francúzsku a Českej republike. Od svojho začiatku sa sústredila na vývoj, implementáciu a predaj špecifického softvéru. Má okolo 60 zamestnancov. V poslednej dobe je hlavným cieľom zmena platformy Delphi a jej nahradenie modernejšou platformou .NET.

Softvér poskytuje zákazníkom modulárny produkt. Výhodou je možnosť ľahko a rýchlo rozšíriť jeho funkcionalitu o ďalšie možnosti presne podľa potrieb zákazníka. Súbor technických nástrojov je poskytovaný každému zákazníkovi individuálne. Stratech vyvíja softvérové projekty, ktoré sú po implementačnej a architektonickej stránke kvalitné a moderné.

Pre vývoj komerčne zameraných projektov kolektív spoločnosti zbiera a spracúva odborné poznatky. Používateľské informácie sa získavajú spätnou väzbou od zákazníkov. Konzultáciou a priamou spoluprácou so spriatelenými firmami Stratech zvyšuje technickú profesionalitu a vlastný potenciál pre vývoji technicky zameraného produktu. [1]

Obrázok 1: Logo firmy Stratech Software

2.1. Pôsobenie na trhu

Pole pôsobnosti spoločnosti je pestré a snaží sa preniknúť do viacerých oblasti na trhu rôznymi softvérovými produktmi. V súčasnej dobe je Stratech dodávateľom pre tieto odvetvia:

1. Rekreačný priemysel
2. Logistika a priemysel
3. Verejná sféra
4. Výskum a vývoj

Rekreačný priemysel

Stratech dlhodobo aktívne spolupracuje s viac ako 250 partnermi v oblasti rekreačného priemyslu. Globálne sa jedná o zautomatizovanie menších technických procesov na rôznych úrovniach firemnej štruktúry. Príkladom sú procesy rezervácie, registrácie, evidencie klientov, a zákaznícke služby. Ďalej poskytujú servis a poradenstvo počas celej doby využívania produktov. [1]

Logistika a priemysel

Už od roku 1989 sa Stratech pohybuje v automatizácii logistiky. Cieľom firmy je pokryť všetky aspekty dokumentácie v medzinárodnej preprave, zoznam pravidiel a poplatkov pri preprave. V priemyselných odvetviach pracuje s výrobnou dokumentáciou. Súčasťou je aj oblasť colných správ. Stratech sa pravidelne zúčastňuje významných priemyselných konferencií, vďaka ktorým firma získava okamžitú reakciu na rôzne návrhy automatizácie a technológie s tým spojené. [1]

Verejný sektor

Dôležitým prvkom firmy je spolupráca s verejnou správou. Snahou je postupne zjednodušiť zložité procesy na úrovni manažmentu a vytvoriť tak systém tvorený jednoduchými časťami. Poskytované produkty nepretržite sledujú aktivitu integračného procesu na strane zákazníka a pomáhajú tak zvýšiť ich efektivitu. Firma zároveň ponúka produkt, ktorý poskytuje dáta pre čerpanie z European Social Fond(ESF). [1]

Výskum a vývoj

V druhej polovici roka 2012 spoločnosť bude poskytovať svojim pilotným zákazníkom nový produkt s názvom RD-Process. Úlohou produktu je celková organizácia manažmentu zameraná hlavne na stredne veľké organizácie, t.j. do 500 zamestnancov. RD-Process poskytuje prostredie pre celkové pokrytie služieb pri vývoji a výrobe nového produktu. Výhodou je aj jeho využitie pri ďalšom vývoji softvéru. [1]

2.2 Česká pobočka

Česká pobočka je mladá spoločnosť. Pri vývoji produktu aktívne spolupracuje so svojou materskou spoločnosťou, ktorá poskytuje celkové know-how a počiatočnú víziu produktu.

Vývojový tým je tvorený 10 zamestnancami. Ich úlohou je starostlivosť o vývoj, implementáciu, predaj a servis komplexného softvéru, ktorý sa používa v oblasti výskumu a vývoja. Špecifickým zameraním firmy sú procesy úzko súvisiace so spracovávaním dát a web hostingom, procesy pri webových aplikáciách a portáloch, poradenstvo v oblasti informačných technológií a informačných systémov obecne. V tíme pracujú prevažne bývalí absolventi VŠB-TUO. Kolektív je mladý, plný energie a veľmi ústretový. Pracovná doba je flexibilná, jeden pracovný deň trvá 8 hodín.

2.3 Pracovné zaradenie

Pri snahe o získanie odbornej praxe bolo nutné zúčastniť sa osobného pohovoru s vedúcim tímu. Bol som informovaný hlavne o technickom zameraní spoločnosti, o používaných technológiách a metódach. Počas druhého pohovoru som sa zoznámil s ďalšími členmi tímu. Mal som možnosť vidieť pracovné prostredie, spôsob komunikácie a technické vybavenie spoločnosti. Osobný pohovor prebiehal v anglickom jazyku s manažérom z holandskej centrály. Cieľom pohovoru bolo zistiť rozsah vedomostí a odborných skúseností s vývojom softvéru.

Prax som vykonával dvakrát do týždňa. Práca mala charakter testovacích úloh, menších implementačných zmien kódu a odstraňovanie chybových hlásení. V prvých dňoch praxe bolo nutné vytvoriť používateľské kontá pre prístup do firemného systému a prejsť úvodným školením. Z dôvodu veľkého rozsahu vyvíjaného produktu som bol zaškolený formou tutoriálov. Osvojil som si zásady práce pod návrhovým vzorom MVVM, zoznámil sa s princípmi akými fungujú jednotlivé komponenty v GUI a vyskúšal som si prácu v TFS 2010. V rámci zoznámenia sa s internou politikou som nadobudol znalosti o správnom zápise kódu (pomenovávanie metód, zachovanie štruktúry kódu apod.) a o dodržiavaní programátorských zvyklostí pri prací pod platformou Prodigy.

Pri práci na zadaných úlohách som bol podporovaný zamestnanci, ktorý boli veľmi ochotný a opakovane mi pomáhali odstraňovať systémové chyby. Ak upravený kód neobsahoval žiadne chyby, nasledovalo porovnávanie s aktuálnym kódom uloženým na serveri. Po úspešnej kompilácii aktuálneho projektu sa upravený kód uložil na server. Týmto sa predchádzalo nekonzistencii kódu počas vývoja.

3. Popis platformy Prodigy

Platforma Prodigy je systém navrhnutý na podporu vývoja a výroby určitého produktu. Tento proces bude optimalizovaný vďaka znalostiam získaním z daného oboru, vývoju trhu a spätnou väzbou od používateľov.

Platforma Prodigy sama sleduje používanie modulov a získané dáta ďalej spracováva vo vývojovom procese, vďaka čomu dosiahneme rýchlejšiu odozvu pri vývoji. Analýzou získaných dát vylepšujeme jednotlivé moduly a postupy, ktoré zefektívnia tvorbu produktu.

Množina nástrojov, ktoré ponúkneme používateľovi pri výrobe produktu sa nazýva modul. Vďaka tomu sa na vývoj produktu pozeráme ako na modulárny proces. Pomocou toho získame logické rozdelenie nástrojov, ktoré pomáhajú pri výrobe a vývine určitého produktu. Dovoľuje nám to podrobnejšie simulovať realitu, rýchlejšie sa prispôsobiť novým podmienkam a kombinovať jednotlivé moduly navzájom.

Obrázok 2: Princíp architektúry platformy Prodigy.

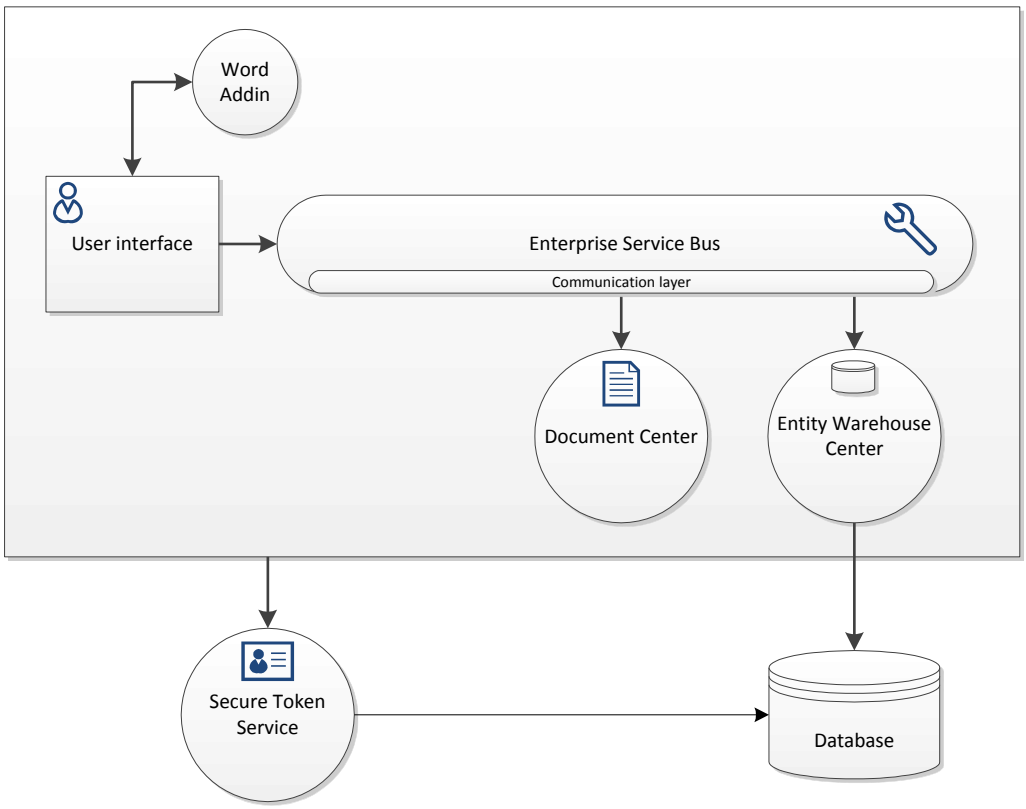
Na obrázku 2 vidíme princíp platformy Prodigy. Vrchné dva cykly predstavujú väzbu medzi základnými súbormi nástrojov a vývojom produktu. Je zrejmé, že táto väzba zohráva dôležitú úlohu v rámci celej platformy. Červený cyklus zobrazuje manažment, ktorý sa stará o uskladnenie a správu základných modulov, riadi ich použitie a koordinuje postup vývoja. Pri

vývoji ďalšieho produktu manažment zaisťuje jeho optimalizáciu pomocou širokej škály používaných modulov.

Platforma Prodigy je atraktívna vďaka zberu dát, optimalizácií modulov a koordinácií vývoja. Táto platforma sa v súčasnosti neustále vyvíja a má za úlohu nahradiť predchádzajúcu, ktorá je založená na jazyku Delphi. Vývoj je zaujímavý používanými technológiami, ktoré sú moderné a user-friendly. [2]

3.1 Architektúra Prodigy

Kompletný súbor produktov ponúkaných firmou je obsiahnutý v Prodigy Software Product Line (SPL), ktorý sa ďalej delí na jednotlivé centrá. Jeho súčasťou je aj platforma Prodigy, ktorej architektúru vidíte na obrázku 3. Platforma sa skladá z troch hlavných komponentov: User Interface (PUI), Enterprise Service Bus (ESB) a Entity Warehouse (PEW), ktoré pokrývajú základné rysy platformy. Ich podrobnejší popis sa nachádza pod obrázkom.



Obrázok 3: Architektúra platformy Prodigy

Prodigy User Interface

PUI je prezentačná vrstva Prodigy a z pohľadu koncového používateľa je táto časť najviac viditeľná. Poskytuje user-friendly prostredie pre komunikáciu s používateľom. Pod komunikáciou chápeme výmenu informácií medzi klientom a ESB. Toto rozhranie je oddelené od logickej časti projektu a zmena grafických komponentov neovplyvní logiku. Inak povedané, slúži ako rozhranie medzi používateľom a biznis logikou. [2]

Prodigy Enterprise Service Bus

ESB má na starosti kompletnú biznis logiku platformy, sprostredkovanie a použitie externých služieb a komunikáciu medzi ostatnými SPL centrami. Logika Prodigy je oddelená od zvyšku kódu. [2]

Prodigy Entity Warehouse

PEW sprostredkováva ESB prístup k dátam. Zodpovednosťou PEW je prístup k databáze, umožňovanie základných operácií nad dátami (uloženie, načítanie) a nad databázou (select, insert, update, delete). Z toho vyplýva, že služby, ktoré majú pracovať s dátami musia prechádzať cez túto vrstvu. [2]

Prodigy Secure Token Service

Ako jediná služba nemusí pri práci s dátami pristupovať k PEW, pretože jej úlohou je zaistiť bezpečnosť celej platformy. To dosahuje dvoma spôsobmi. Prvým spôsobom je, že STS zástava úlohu licenčného manažéra. Dozerá na licencie zákazníkov a zaručuje, že žiadny zákazník nebude mať prístup k modulom, ktoré pre neho nie sú povolené. Jednoduchým spôsobom kontroluje interné údaje o vydaných licenciách s tými, ktoré používajú zákazníci a garantuje vždy konzistentné správanie prístupu k službám a modulom.

Druhý spôsob zabezpečenia Prodigy je, že funguje na princípe bezpečnostného tokenu (hardvérové alebo softvérové zariadenie) . Tento token je poskytnutý používateľom a využíva princíp digitálneho podpisu. Pomocou neho prebieha autentizácia a autorizácia do všetkých centier platformy Prodigy ale aj mimo nej. [2]

Prodigy Document Center

PDC sa používa ako služba pracujúca s dokumentmi v rámci platformy a okrem iného umožňuje ukladať, načítať a upravovať dokumenty, šablóny a ďalšie elektronické materiály používané v rôznych centrách Prodigy. [2]

3.2 Používané technológie

Snahou firmy je pri vývoji používať najmodernejšie technológie a osvedčené postupy pomocou návrhových a architektonických vzorov, ktoré obsahujú známe a efektívne riešenia určitých programátorských problémov. Vďaka tomu som si pri práci s produktom vyskúšal dané technológie a získal predstavu o ich funkčnosti v rámci celého projektu.

3.2.1 Microsoft .NET Framework (.NET)

Microsoft .NET je kolekcia softvérových technológií (platforma) z dielne Microsoftu. Platforma je navrhnutá pre zjednodušenie vývoja aplikácií. Jej hlavnou prednosťou je vytváranie, spúšťanie aplikácií a XML webových služieb novej generácie. Hlavné zložky systému tvoria Spoločný jazykový modul runtime (CLR - Common Language Runtime) a Knižnica tried .NET Framework (BLC - Basic Class Library).

CLR je základom rozhrania .NET. Jeho úlohou je správa kódu v dobe spúšťania aplikácie, správa pamäti a vlákien. Zaisťuje prísnu bezpečnosť typov spolu s ďalšími službami, ktoré podporujú zabezpečenie a robustnosť.

Knižnica tried je objektovo orientovaná kolekcia opakovane používaných typov, ktoré môžeme v aplikácii využiť. Podporuje tradičné aplikácie pre príkazový riadok, grafické užívateľské rozhranie (GUI) a technológie ASP.NET (webové formuláre alebo XML webové služby). Komponenty tretích strán môžu spolupracovať s triedami v rozhraní .NET. Najnovšia verzia je .NET Framework 4.5. [3]

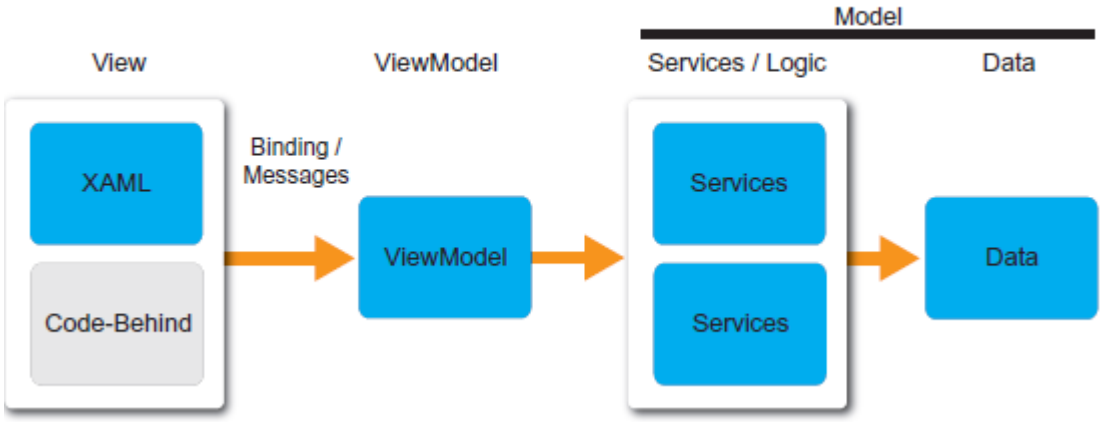
3.2.2 C# (C-sharp)

C# je objektovo orientovaný programovací jazyk, ktorý vytvorila firma Microsoft spoločne s platformou .NET. Vyšla ako súčasť programu Visual Studio v roku 2000. Využitie našiel pri tvorbe formulárových a webových aplikácií, webových služieb a databázových programov.

S postupným tvorením nových verzií .NET pribúdali aj nové možnosti jazyka. Príkladom sú statické triedy, generiká, iterátory, indexeri a anonymné metódy pre použitie v delegátoch. Preto môžeme konštatovať, že tento jazyk je zjednodušená, vylepšená a čisto objektová verzia programovacieho jazyka C++. [4]

3.2.3 Návrhový vzor Model-View-ViewModel

Pri vývoji na platforme Prodigy sa často používa návrhový vzor Model-View-ViewModel, tzv. MVVM. Pomocou tohto návrhového vzoru získame oddelenie vlastného pohľadu (View) a logiky (ViewModel). Vďaka tomu ubudne kód z code-behind. Oddelenie logiky a pohľadu nám pomáha v internej organizácii jednotlivých časti celej aplikácie a rozdeľuje aplikačnú logiku a UI do viacerých vrstiev. Výhodami je dobrá použiteľnosť, testovateľnosť a udržateľnosť kódu.



Obrázok 4: Architektúra návrhového vzoru Model-View-ViewModel

Stručné rozdelenie MVVM návrhového vzoru :

- **Model** – má na starosti úložisko dát, s ktorými pracujeme a stará sa o biznis logiku. Manažment koordinuje biznis logiku nad aplikačnými dátami, ktoré sú vopred pripravené (validované). Model môže obsahovať entity a služby napr. webové služby, biznis služby, prístup k dátam atď.
- **View** – je to XAML súbor spoločne s jeho code-behind. Jeho úlohou je interakcia s používateľom. Kód, tu uložený, je logickou časťou View (riadiace interakcie medzi View elementmi alebo aplikáciami). View obsahuje dostatok informácii o štruktúre

ViewModel-u nutnej k bindovaniu (spôsob komunikácie) ale o zvyšku systému nemá informácie.

- **ViewModel** – jeho úlohou je prispôsobiť dáta Model-u pre zobrazené View. Obsahuje kód logiky na prezentačnej vrstve. View sa pomocou databinding-u odkazuje na jeho verejné vlastnosti. Obsahuje definíciu príkazov, t.j. akcií, ktoré môžeme z View volať. [5]

3.2.3 Microsoft Silverlight

Táto technológia od spoločnosti Microsoft je určená pre tvorbu dynamického obsahu webových stránok, ktoré môžu kombinovať klasické textové prvky, vektorovú a rastrovú grafiku, animácie a video. Súhrne sa takéto webové služby označujú RIA aplikácie. Najnovšou verziou je Silverlight 5.1, ktorá ponúka podporu webovej kamery, mikrofónu a „Rich text editor“, ktorý zabezpečuje rozšírené možnosti formátovania textu. Paralelne s touto technológiou bola vyvíjaná aj ďalšia technológia Windows Presentation Foundation (WPF), ktorá bola určená predovšetkým na desktopové aplikácie (aplikácie na pracovnej ploche). S treťou verziou sa rozdiely medzi týmito platformami takmer vytratili a Silverlight mohol byť použitý, na rozdiel od WPF aj na iných platformách pomocou pridania jednoduchého doplnku. [5]

3.2.4 Microsoft Prism

Jedná sa o projekt, ktorý založil Microsoft. Jeho snahou je podchytiť efektívne používanie osvedčených návrhových vzorov a programovacích praktík pri vývoji aplikácií vo WPF alebo Silverlight. Prism je nazývaný ako príručka pre kompozitné aplikácie. Presnejšie môžeme povedať, že je to súbor platforiem (napr. Component Application Library – CAL), návrhových vzorov (Inversion of Control, Separated Presentation), vďaka ktorým naša aplikácia bude vytvorená rýchlejšie a obecne zjednoduší vývoj. Aktuálnou verziou je Prism v5. [6]

3.2.5 Windows Communication Foundation

WCF je jednotná platforma, ktorú priniesol .NET Framework 3.0 a je určená na vytváranie servisne orientovaných aplikácii (SOA). Platforma obsahuje Microsoft technológie používané pri vytváraní distribuovaných aplikácií. Základným prvkom sú služby komunikujúce

cez koncové body (jeden alebo viac), ktoré prijímajú a odosielajú odpovede.

[5]

3.2.6 Managed Extensibility Framework

MEF je knižnica predstavujúca kompozitnú vrstvu pre tvorbu ľahko rozšíriteľných aplikácií v .NET Framework 4.0. Umožňuje prezerať a používať rozšírenia bez nutnej konfigurácie a poskytuje jednoduchú cestu k zapuzdreniu kódu. Táto technológia preto vytvára modulárne rozšíriteľné aplikácie. To dosiahneme použitím architektonických princípov a návrhových vzorov (napr. Inversion of control, Depedency injections). Princíp celej technológie je spárovanie závislých a závisiacich komponentov (kľúčové slová: import, export).

[6]

3.3 Metodika vývoja

Stratech pri vývoji softvéru využíva SCRUM metodiku. Jedná sa o agilnú (aktívnu) metodiku, v ktorej vývoj prebieha v iteráciách. Ich jednotlivá časová dĺžka sa líši od konkrétnych funkčných požiadaviek a náročnosti danej aplikácie. Každá iterácia má v sebe funkčné požiadavky, ktoré by mal výsledný kód obsahovať.

Jednotlivé iterácie danej aplikácie tvoria časovo a logicky oddelené celky, ktoré napĺňajú všetky funkčné požiadavky. V rámci každej iterácie sú určité špecifické úlohy, ktoré sa behom času menia, vyvíjajú, špecifikujú alebo pribúdajú a umožňujú prehľadnejšiu predstavu o vývoji. Tieto úlohy sa podľa náročnosti ohodnotia približným časovým úsekom a rozdelia do rôznych oblastí (architektúra, implementácia, testovanie).

Každý deň majú členovia tímu stretnutia, ktorých obsahom je:

- Zhodnotenie progresu, ktorý daný člen tímu vykonal od posledného stretnutia. Výstupom je funkčný kód, ktorý sa podarilo bezchybne skompilovať so zvyškom kódu a nahrať na server.

- Zhodnotenie riešenia a finálnej implementácie úlohy, diskusia o výskyte rôznych komplikácií, návrhy k zlepšeniu a celkové zhodnotenie náročnosti danej úlohy.

- Diskusia o plánovaných úlohách, ich podrobný popis a bližšie informácie o konkrétnom postupe pri riešení.

Pri používaní SCRUM metodiky je dôležité aj postavenie tím lídra. Jeho úlohou je organizovať kolektív počas vykonávania práce a dohliadať na dodržiavanie stanovených termínov pridelených jednotlivým úlohám.

Výhodou je možnosť výberu úlohy podľa vlastného uváženia. Na rozdiel od iných metodík (napr. vodopádový model) SCRUM garantuje, že sa chyby odhalia "za pochodu" a tak umožňuje opraviť chybný kód.

3.4 RD-Process

RD-Process predstavuje systém pre riadenie produktového manažmentu, tzv. product management. Tento systém bude podporovať výskum a vývoj produktov pri evidencií a manipulácií s dátami. Vývoj produktov bude mať charakterovo rovnaké procesy ako napríklad obstarávanie materiálov, dovoz techniky, samotnú výrobu produktu a prácu so zákazníkmi. Hlavnou ideou je, že systém umožní vyvinúť produkt prechádzajúci všetkými fázami, t.j. od myšlienky cez plánovanie až do úspešného záveru.

RD-Process je založený na platforme Prodigy a skladá sa z týchto základných časti: Documents, Glossary, Maintenance, Products, Projects, Stakehoders a SupportTables. Moduly Customer Relation Module (CRM) a Human Resources Module (HRM) rozšíria produkt o ďalšie funkčné možnosti.

Documents má v systéme zásadnú úlohu, keďže tvorí úložisko pre celkovú dokumentáciu, ktorá je pri vývoji rozhodujúca. Tieto dokumenty môžu byť technického (výkresy, šablóny či technické návody) alebo obchodného charakteru (účtovníctvo a podklady k uzatvoreným transakciám).

Glossary spravuje používané skratky a kľúčové slová, ktoré sa počas životného cyklu daného produktu používajú.

Modul Products má na starosti popísať a presne definovať vyvíjaný produkt pomocou štandardnej a produktovej časti. Každá z nich musí byť podrobne definovaná a presne určená čo bude vykonávať. Použijeme use-case a activity diagramy, ktoré nám pomôžu lepšie pochopiť požiadavky kladené na produkt.

Stakeholders je modul popisujúci všeobecne tretiu stranu, ktorá ma vplyv na vývoj produktu.

Projects nám priblíži reálny vývoj produktu. Tento modul rozdelí vývoj na menšie časti, projekty, ktorými sa budeme zaoberať lokálne a v danom čase. Skladá sa z RDProjects, ktorý spoločnosť sama vyvíja a StakeholderProjects, ktorého vývojom sa zaoberá tretia strana.

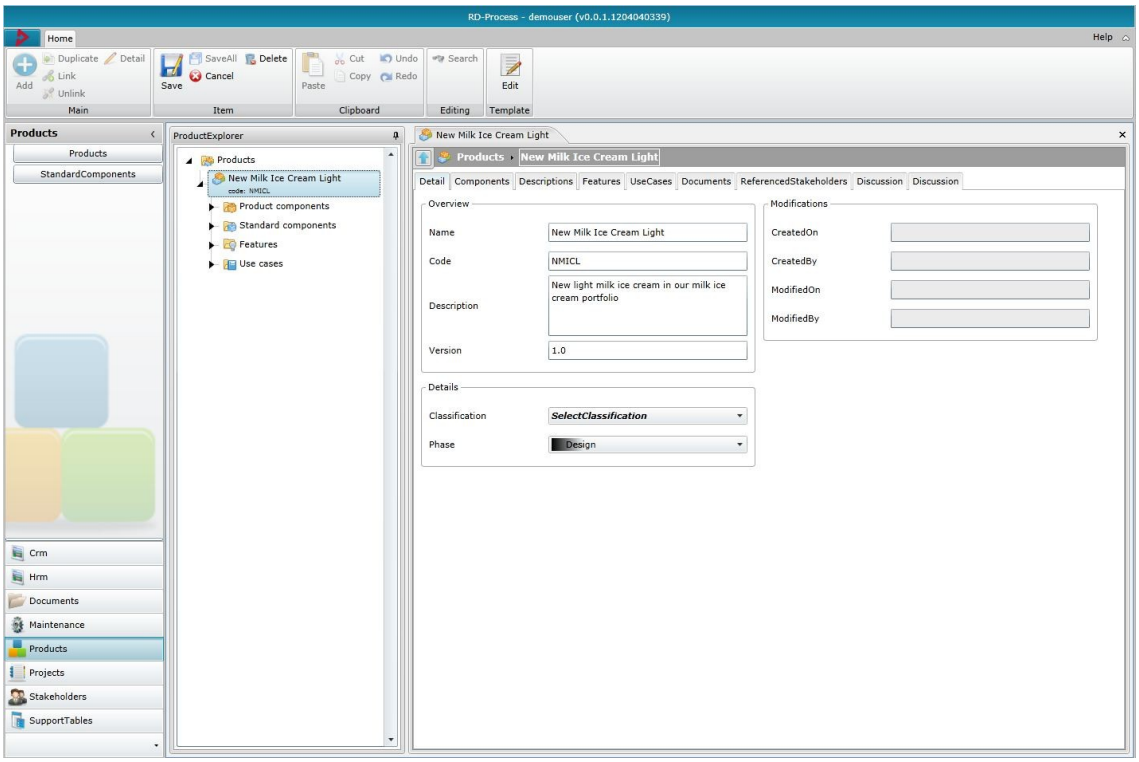
Maintenance a SupportTables nám umožňujú filtrovať rôzne projekty a produkty podľa zvolených kritérií.

3.5 Popis vybraných úloh

Mojou prvou úlohou bolo testovanie vybraných modulov v UI (Products, Projects), viď obrázok 5. Počas tejto úlohy som musel v jednotlivých modulov testovať vybrané možnosti a zapisovať prípadné chyby, ktoré som následne ohlásil vývojárom. Medzi kontrolovanú funkcionalitu patrilo napr. overovanie korektného pridávania a mazania položiek, zobrazenie a správanie View a ViewModel-u v rámci jednotlivých častí modulu. Ďalej kontrola jednotlivých položiek v menu a ich reakciu na pravé stlačenie myši po celej ploche objektu apod.

Neskôr ako súčasť úlohy pribudla kontrola vykreslenia ScrollBar-u pri zmenšení View v module. Úloha bola o to zaujímavejšie, že chýbajúce ScrollBar-y som musel dopísať do projektu. Zaznamenal som si meno modulu, jeho View a našiel danú triedu v projekte. Išlo o úpravu View, to znamená, že všetky úpravy som uskutočňoval v XAML súbore. Pridaním prvku ScrollViewer a nastavením jeho atribútov HorizontalScrollBarVisibility a VerticalScrollBarVisibility na hodnotu “Auto”. Tým som dosiahol to, že pri zmene rozmeru okna a zalomeniu textu sa automaticky objaví ScrollBar. Túto úpravu bolo nutné pridať do každého View. Časom som začal využívať možnosť štýlu (kolekcia hodnôt vlastností, ktoré sa aplikujú na požadované elementy). Na úrovni projektu som vytvoril štýl, ktorý definoval dané vlastnosti ScrollBar-u a v jednotlivých View sa staticky načítal štýl z Resources. Použitím štýlu som výrazne zefektívnil prácu algoritmu.

Vypracovanie úlohy mi trvalo asi 5 dní. Počas nich som pridal do celého projektu približne 900 riadkov kódu a zaznamenal približne 130 chýb. Úloha mi pomohla pochopiť UI a jeho komponenty.



Obrázok 5: Ukážka modulu Products v RD-Process

Ďalšou komplexnejšou úlohou bolo zmeniť implementáciu funkčného kódu. Úlohou bolo zmeniť logiku práce jednotlivých komponentmi modelov, čím sme dosiahli vyššiu úroveň abstrakcie systému. Modelom myslím napr. RDPMaintenance, ktorý obsahuje aj RDPSupportTablesRepository. Každý komponent mal množinu metód, ktoré pracovali s dátami, nastavovali, získavali záznamy podľa atribútov, overovali existenciu záznamov atď. Cieľom bolo, aby výsledný kód mal logiku práce uloženú na jednom mieste.

V prvom kroku som zistil aké ma daný komponent metódy a podľa toho som vytvoril jeho rozhranie, napr. IStakeholderRepository v module RDPStakeholderManagement. Následne som vytvoril ďalšiu triedu, ktorá dedila z daného rozhrania a implementovala jeho metódy. Vytvoril

som logger pre daný typ, ktorý sledoval priebeh metódy a jednotlivé dátové operácie boli vykonávané pomocou abstraktnej triedy DataServiceQuery.

```
private static readonly Logger _Log = Logger.GetLogger(typeof(StakeholderRepository));
public IList<Stakeholder> GetStakeholders()
{
    var context = new DataServiceContextEx(AppSetting.PewUri);
    IList<Stakeholder> stakeholders = null;
    try
    {
        var query = (DataServiceQuery<Stakeholder>)from d
                                                   in context.CreateQuery<Stakeholder>(Stakeholder.EntitySetName)
                                                   select d;
        stakeholders = query.Execute().ToList();
        if (stakeholders == null)
        {
            throw new DataServiceRequestException("result of get data operation should not be null");
        }
        _Log.Debug("Number of fetched Stakeholders: {0}", stakeholders.Count);
    }
    catch (DataServiceRequestException exception)
    {
        _Log.Debug("Get Stakeholders failed: {0}", exception.Message);
    }
    return stakeholders;
}
```

Obrázok 6: Príklad vytvorenej metódy GetStakeholder

Volanie týchto metód mala na starosti servisná vrstva, viď obrázok 7. V tejto triede som musel pre každú metódu vytvoriť dve metódy (prefix Begin a End), pričom prvá vracia status asynchrónnej operácie a druhá vracia výsledok databázovej operácie (napr. objekt Stakeholder). V metóde Begin som vytvoril objekt pre daný komponent, ktorý volal špecifickú metódu a vracal typ napr. IList<Stakeholder>. Výsledný kód bol prehľadnejší, efektívnejší a umožňoval jednoduchšiu prácu s metódami modelu. Vypracovanie úlohy zabralo vyše 15 dní a pre každý komponent som napísal 600-1300 riadkov kódu, podľa počtu metód. Počet komponentov bol približne 40. Pre lepší prehľad ponúkam v prílohe 1 príklad vypracovaných súborov.

```
public IAsyncResult BeginGetStakeholderById(long idStakeholder, AsyncCallback callback, object state)
{
   var repository = new StakeholderRepository();
   var task = Task<Stakeholder>.Factory.StartNew(_ => repository.GetStakeholderById(idStakeholder), state);
   if (callback != null)
   {
      task.ContinueWith((result) => callback(task));
   }
   return task;
}
public Stakeholder EndGetStakeholderById(IAsyncResult result)
{
   var task = result as Task<Stakeholder>;
   if (task == null)
   {
      throw new InvalidOperationException("task should not be null");
   }
   return task.Result;
}
```

Obrázok 7: Fragment kódu triedy StakeholderService

Po vyriešení predošlých úloh som dostal náročnejšie zadanie. Jeho cieľom je upraviť logiku správania sa ChildWindow a spôsob návratu dát späť do hlavného ViewModel-u, ktorý vyvolal daný ChildWindow. ChildWindow je v tomto prípade vyskakovacie okno, ktoré sa zavolá pri pridávaní alebo mazaní komponentov daného View. Úloha vyžadovala širšie znalosti s návrhovým vzorom MVVM. Pracoval som v konkrétnych moduloch Maintenance, Products a Stakeholders. V predošlej implementácii sa o túto funkcionalitu staral EventAggregator, ktorý ošetroval jednotlivé udalosti.

Mojou prácou bolo najprv inicializovať ChildWindow pomocou preťaženia metódy inštancie InteractionRequest, ktorá vracala metódu zavolanú pri zatvorení ChildWindow. Dané View obsahovalo štandardnú definíciu používateľského trigger-u, ktorý sa využíva pri práci s IInteractionRequest objektmi. V code-behind View daného ChildWindow sa nenachádzal žiaden kód (žiadne udalosti, registrácie apod.) okrem Export atribútov. V tomto View bolo nutné pridať do XAML súboru definíciu ChildWindow a jeho tlačidiel (OK, Cancel).

ViewModel bol obohatený o vlastnosť Confirmed, ktorá vracala typ bool a volala sa pri zmene View. Ďalej som upravoval metódu, ktorá sa volala v hlavnom ViewModel-i a ako parameter som zadal špecifický ChildWindow ViewModel (napr. ProductClassificationViewModel) objekt, ktorý dostával spätnú väzbu z ChildWindow. Tento objekt som následné otestoval podmienkou

objekt.Confirmed.HasValue && objekt.Confirmed.Value !=null

aby som zaručil, že pri kliknutí na Cancel sa nebude vykonávať žiadna operácia. Ak podmienka prešla, vrátili sa dáta z ChildWindow a spracovali sa v danom ViewModel-i. Často išlo o kolekciu objektov, ktoré chce používateľ pridať alebo odstrániť.

Výsledkom tejto zmeny bol jednoduchší a prehľadnejší kód (žiadne udalosti v code-behind daného ChildWindow), väčšia kontrola nad vykonávaným procesom a pridanie štandardnej funkcionality Prism. Tato úloha bola značne náročná, pretože určité triedy boli v projekte zle pomenované a nájsť správny ViewModel pre dané View bolo často krát časovo náročné. Úlohu som preto vykonával 9 dní a počas nich som napísal 1000-1300 riadkov kódu.

4. Prínosy bakalárskej praxe

Ako hlavný prínos považujem možnosť nahliadnuť do vývoja softvéru. Jednotlivé kroky uceleného procesu, či už návrh vývoja, vývoj samotný alebo testovanie projektu spolu úzko súvisia. Ako laik som dostatočne nepochopil náročnosť a robustnosť týchto krokov.

Počas odbornej praxe sa môj záujem o vývoj softvéru výrazne prehĺbil. Jednotlivé aspekty vývoja, ktoré mi prišli nezaujímavé a nepotrebné sa naopak ukázali ako veľmi atraktívne a nenahraditeľné v celkovom procese vývoja produktu.

4.1 Získane znalosti a odhalené nedostatky

Kolektív firmy mi pomohol pochopiť dôležitosť tímovej práce. Hlavne pri pochopení náročnejších problémov alebo pri samotnom programovaní kde je praktické ak sú kolegovia blízko seba a majú priamy kontakt medzi sebou. Komunikácia je týmto rýchlejšia a pomáha tak ostatným pochopiť tie časti systému, s ktorými nepracujú alebo sa v danej časti projektu nevyskytujú.

Pri práci na úlohách som pochopil, že architektúra projektu a celého produktu je rozhodujúca pre konečnú podobu kódu. Dôležité je mať pred samotným vývojom správne navrhnutú softvérovú architektúru aby sa počas neho nemusela meniť. Každá zmena v architektúre vynúti zmenu v kóde a pri veľkých projektoch môžu tieto zmeny trvať radovo niekoľko týždňov.

Počas plnenia zadaných úloh som často vyhľadával pomoc na oficiálnych stránkach Microsoft MSDN. Vzhľadom k tomu, že text bol v anglickom jazyku bolo nutné ovládať technické pojmy. To ma motivovalo aby som ďalšie publikácie o softvérovej tematike čítal po anglicky.

Hlavné nedostatky som objavil pri riešení algoritmu úloh. Kód bol často pridlhý a zbytočne komplikovaný. Išlo o neznalosť efektívneho zápisu algoritmov a prepojenie programu s inými časťami projektu. Kolegovia mi ukázali ako efektívne pracovať vo vývojovom prostredí Visual Studio pomocou skratiek a programov, ktoré zvyšujú výkon pri programovaní, prípadne automatizujú určité operácie (napr. generovanie metód podľa názvu metódy atď.).

4.2 Hodnotenie bakalárskej praxe

Počas bakalárskej praxe som sa naučil množstvo nových informácii ako používať vývojové nástroje a pracovať na reálnom softvérovom projekte. Z daných dôvodov hodnotím odbornú prax pozitívne. Kvalita mojich riešených úloh nebola dostačujúca v porovnaní s prácou ostatných zamestnancov. Príčinou bola nesúvislá pracovná doba (dvakrát do týždňa) a nedostatok odborných znalostí (programovacie techniky, neznalosť podporných programov atď.). Prínosom je zlepšenie mnou navrhovaných riešení a samotného programovania. Hlavným cieľom je riešiť problémy efektívne a jednoducho. Práca s ostatnými zamestnancami bola motivujúca a získané odborné poznatky zvýšia moje šance pri hľadaní práce v tomto odbore.

5. Literatúra

[1] Stratech Software. [online]. 2012. Dostupné z: www.stratech.nl

[2] QUINTEN, Vincent. STRATECH SOFTWARE. *PDY-MI-39-Prodigy Architecture Guide.* 2011, 73 s.

[3] PETZOLD, Charles. *.NET Book Zero*. Petzold, Charles. Version 1.1, 2007, 5 s ISBN-10 0126051421

[4] SMITH, A. Rod a NORDLUND, Jonas. *C# Programming*. Wikibooks, Version 1.2, 2007

[5] BROWN, Pete. *Silverlight 4 in Action: Silverlight 4, MVVM, and WCF RIA Services*. Greenwich: Manning, 2010, ISBN 19-351-8237-4

[6] BRUMFIELD, Bob. *Developer's guide to Microsoft Prism 4: building modular MVVM applications using Windows Presentation Foundation and Microsoft Silverlight.* Redmon, Wash: Microsoft, 2011, ISBN 07-356-5610-X

6. Zoznam príloh

I. Príloha na CD

1. Vypracovanie bakalárskej práce
2. Príloha č.1